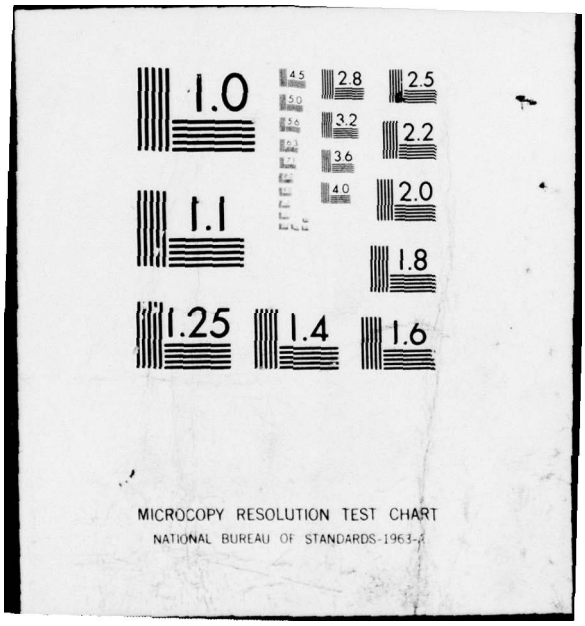
AD-A040 066 YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE F/G 13/8
APPLICATION OF SPARSE MATRIX TECHNIQUES TO RESERVOIR SIMULATION--ETC(U)
1975 P T WOO, S C EISENSTAT, M H SCHULTZ N00014-67-A-0097-0016

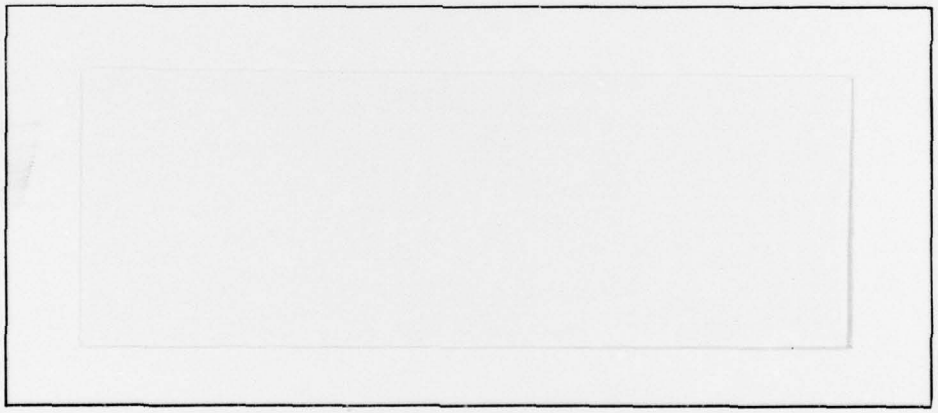
UNCLASSIFIED RR-53 NL

| OF |
AD
A040066

END

DATE
FILMED
6-77

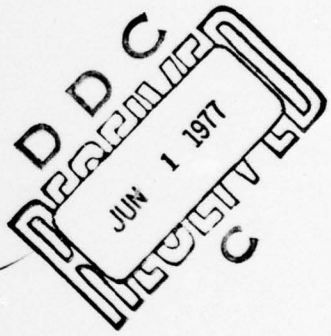

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

P. T. Woo is at Chevron Oil Field Research Company, La Habra, California 90631.

A. H. Sherman is at the Department of Computer Science, University of Illinois, Urbana, Illinois 61801.



(6) Application of Sparse Matrix Techniques to Reservoir Simulation.

(10) P. T. Woo, S. C. Eisenstat, M. H. Schultz, and A. H. Sherman

(9) Research Report, #53

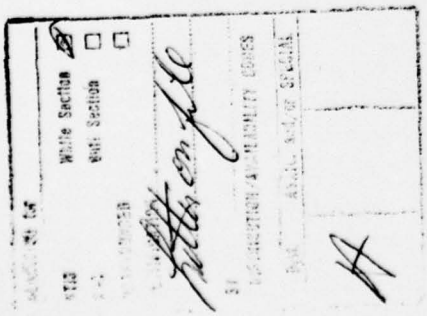
(11) 1975

(14) RR-53

(12) 14p.

(15) N00014-[redacted]-67-A-0097-0016.
NR-044-401
UILU-432

This work was partially supported by ONR Grant [redacted]-67-A-0097-0016.



407451

Introduction

The purpose of this paper is to show how sparse Gaussian elimination is applied to the numerical simulation of petroleum reservoirs. Our emphasis will be on this particular application, somewhat in the style of [10], rather than on theoretical or implementation questions, which are treated in other papers in this series, [3] and [4]. In particular, we shall present the work and computing-time requirements of sparse Gaussian elimination for some typical problems of reservoir simulation.

In many reservoir simulation problems, we seek the solution of a system of nonlinear parabolic partial differential equations describing multiphase flow in two or three space dimensions. In this paper, we restrict our attention to two-dimensional problems.* The most common technique is to approximate the domain by a rectilinear mesh or grid and to approximate the partial differential equations by five-point difference equations together with suitable linearizations. The result is a sequence of systems of linear equations

$$A x = b \tag{1}$$

where entries in the matrix A and the vector b vary from time step to time step. A simulation problem may involve up to a thousand or more such systems.

In reservoir simulation, systems of the form (1) have usually been solved with iterative rather than elimination methods. This was thought to save both time and storage. But selecting an efficient iterative method, optimal acceleration parameters, and a good stopping criterion is sometimes difficult and expensive. Moreover, recently we have found that in some situations iterative methods will not converge to an acceptable solution within a reasonable number of iterations because of the increasing complexity of simulation problems.

Classical elimination methods, though inefficient in terms of both speed and

---

* See [8] and [10] for computational results for coarse three-dimensional grid problems. Sparse Gaussian elimination for fine three-dimensional grids is prohibitively expensive [2,3].

storage, have avoided these difficulties and have always yielded satisfactory solutions. Recently, sparse matrix techniques have greatly improved the computing speed and storage efficiency of elimination methods, and they have become an important tool in reservoir simulation.

The computing mesh in reservoir simulation is customarily numbered row by row or column by column (the grid row ordering). Depending on the derivation of the difference equations, the system (1) can be in one of the two forms described below.

Type 1: In this case, x represents unknown reservoir pressures at the grid points. A is a diagonally dominant band matrix, which is usually nonsymmetric, although its incidence matrix is symmetric. As an example, a 3×4 grid and a corresponding matrix A are shown in Figure 1.

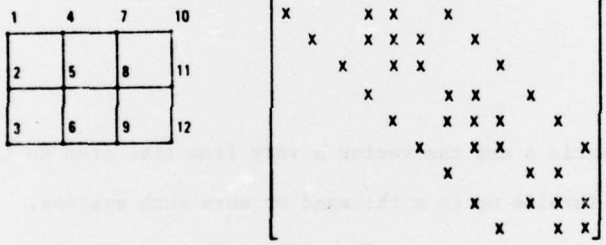


Figure 1:
Type 1 Matrix from a 3×4 Grid.

Type 2: In this case, x represents unknown reservoir pressures and/or fluid saturations (volumetric fractions) at the mesh points. If the unknowns at each grid point are numbered consecutively, the elements of A cluster in blocks as in Figure 2. Some of the matrix entries within each block may be zero. Both A and its incidence matrix are usually nonsymmetric, and A is not always diagonally dominant, although pivoting is not required for numerical stability. This type of matrix will be referred to as a block matrix.

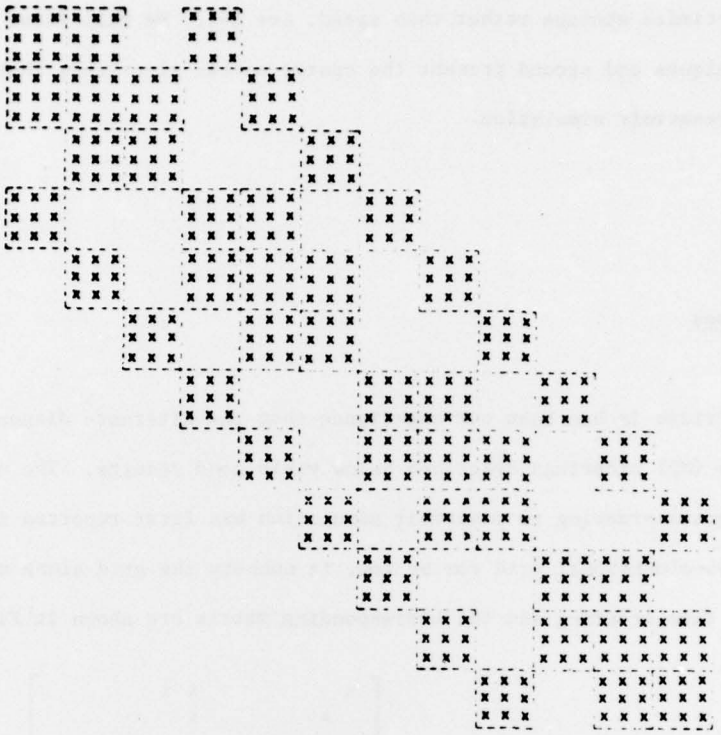


Figure 2:
Type 2 (Block) Matrix from a 3×4 Grid

The application of sparse matrix techniques to reservoir simulation consists of two steps. First, we renumber or reorder the grid to minimize the fill-in during elimination and to minimize the number of arithmetic operations involving nonzeroes. This reordering is equivalent to a symmetric permutation of the matrix A of the form PAP^t where P is the permutation matrix corresponding to the reordering of the grid. Second, we solve the permuted system using Gaussian elimination where we store and operate on only the nonzeroes of A plus the new nonzeroes that occur during elimination. In our experience with reservoir simulation, the grids are relatively coarse because of the lack of detailed geological data describing the reservoir or because of the need for only "engineering" estimates of the unknowns. Hence, we will assume that in-core storage is not a critical issue and will emphasize an implementation designed for maximum efficiency with respect to computing time rather than computer storage. For a discussion of related implementations

designed to optimize storage rather than speed, see [4]. We will first describe the ordering techniques and second present the sparse matrix algorithms that we have found effective in reservoir simulation.

Ordering Schemes

For Type 1 matrices is has been our experience that the alternate diagonal (AD) and the minimum degree (MD) orderings described below yield good results. The application of the alternate diagonal ordering to reservoir simulation was first reported in [8]. Instead of numbering a two-dimensional grid row by row, it numbers the grid along alternate diagonals. An example of this ordering and the corresponding matrix are shown in Figure 3.

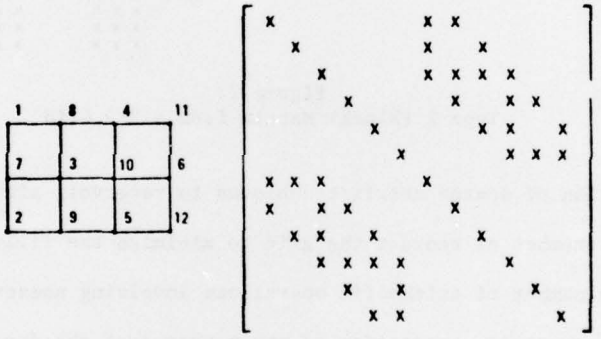


Figure 3:
Alternate Diagonal Ordering and Its Type 1 Matrix

The application of the minimum degree (MD) ordering scheme to reservoir simulation was discussed in [10]. The general idea is that at any stage of the elimination procedure we select as pivot row that row with the fewest nonzero off-diagonal elements. In case of a tie, we select any row from the tie. In the minimum degree ordering, we can start with a matrix that corresponds to either the grid row ordering or the alternate diagonal ordering. (The only effect is to modify the breaking of ties.) The latter sometimes

results in as many as 15% fewer multiplications than the former.

For Type 2 matrices, we have found that it is convenient and effective to number the grid or permute the matrix blocks according to one of the schemes above. The relative positions within each block are unchanged.

Work Requirement

We define the work requirement as the number of multiplications and divisions required to solve the system (1) by sparse elimination. For Type 1 linear systems, the work requirement of the alternate diagonal ordering was given in [8]. For large two-dimensional grids with dimensions I and J ($I \geq J$), the work requirement with AD ordering is

$$W_{AD} \simeq \frac{IJ^3}{2} - \frac{J^4}{4} \tag{2}$$

while for large I×I squares it is

$$W_{AD} \simeq \frac{I^4}{4} . \tag{3}$$

The AD ordering is not asymptotically optimal, since the work required with an optimal ordering scheme is $O(I^3)$ for two-dimensional square grids [6]. In practice, however, the AD ordering works fairly well.

We do not know of any formulas for calculating the work required with the minimum degree ordering. The observed work requirement for several typical problems will be given later. This ordering seems to work as well as or better than the alternate diagonal ordering.

The work requirements with various orderings for some rectangular grids are given in Table 1. GR, AD, and MD designate respectively the grid row, alternate diagonal, and minimum degree orderings. We observe from this table that reordering the grid substantially reduces the work requirement, with the greatest reduction for square grids.

Grid Dimensions	Ordering	Row Algorithm	Grid Dimensions	Ordering	Row Algorithm
10×10	GR	11,696	10×20	GR	24,896
	AD	5,504		AD	13,504
	MD	4,936		MD	13,562
30×30	GR	855,096	20×60	GR	541,796
	AD	261,104		AD	255,604
	MD	189,276		MD	232,372
50×50	GR	6,458,496			
	AD	1,809,504			
	MD	1,149,772			

Table 1:
Work Requirements for Type 1 Equations

Reordering is less effective in reducing the work requirement of elongated rectangular grids, though such grids require considerably less work than square grids with the same number of grid points.

Table 2 shows the work requirement of three two-dimensional grids from actual reservoir simulation problems. These grids are characterized by irregular boundaries as shown in Figures 4, 5, and 6. Clearly reordering is also effective in reducing the work

Simulation Problem	Grid Dimensions	Number of Equations	Work Requirement		
			GR	AD	MD
1	8×69	390	20,726	15,218	16,110
2	23×37	507	174,974	62,430	51,766
3	55×72	2,347	~7,200,000	2,037,432	709,442

Table 2:
Work Requirements for Type 1 Equations
Simulation Problems 1, 2, and 3

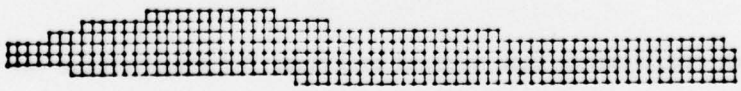


Figure 4:
Simulation Problem 1, 8×69, 390 Equations

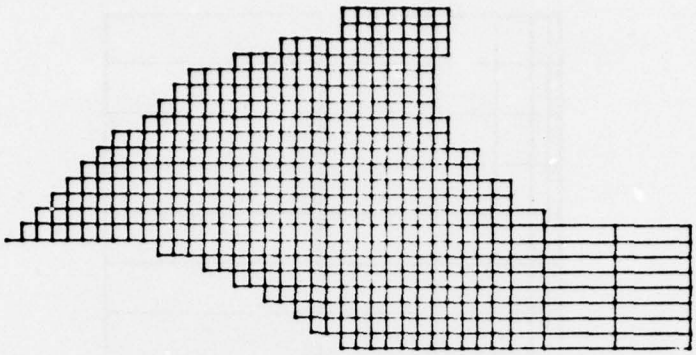


Figure 5:
Simulation Problem 2, 23×37, 507 Equations

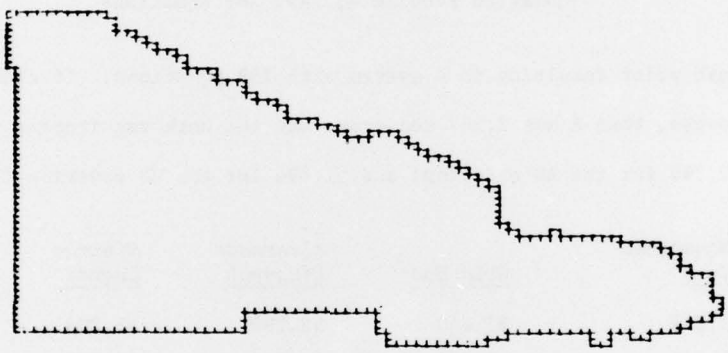


Figure 6:
Simulation Problem 3, 55×72, 2,347 Equations

requirement for such grids. In simulation problem 3, the AD ordering reduces the work requirement by a factor of about 4, and the MD ordering reduces it by a factor of about 10.

For Type 2 systems, suppose there are k unknowns per mesh point. The simplest way of applying sparse matrix techniques to the system (1) is to assume that each block of matrix A has dimensions k×k and contains k^2 nonzeroes. The work requirement for a Type 2 matrix in this simple situation is then approximately equal to k^3 times that of the Type 1 matrix based on the same ordering.

As an example, we give the work requirement for a block matrix A arising from reservoir simulation problem 4 with a 7×9 grid as shown in Figure 7. There are three

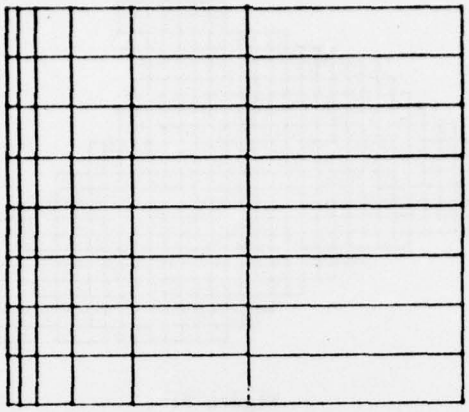


Figure 7:
Simulation Problem 4, 7×9, 189 Equations

unknowns per mesh point resulting in a system with 189 equations. If each block of A is assumed to be dense, then A has 2,547 nonzeroes and the work requirement is 91,640 for the GR ordering; 52,290 for the AD ordering; and 51,894 for the MD ordering (Table 3).

Nonzeroes in A	*Grid Row*	*Alternate Diagonal*	*Minimum Degree*
2,547	91,640	52,290	51,894
1,061	15,213	11,611	17,767

Table 3:
Work Requirements for Simulation Problem 4

Actually each block of A may contain fewer than nine nonzeroes. In this example, A has only 1,061 actual nonzeroes at the start of the simulation, and the number gradually increases to 2,547 nonzeroes in the limit, as the number of time steps becomes very large. The work requirement corresponding to 1,061 nonzeroes in A is 15,213 for the GR ordering, 11,611 for the AD ordering, and 17,767 for the MD ordering (minimum degree with respect to the grid). By using a good grid ordering and taking into account all the zeroes in A, the work requirement is substantially reduced from 91,640 to 11,611. Thus, it is advantageous to account for the change in the number of nonzeroes in A during simulation, when an

efficient code is available to do so.

Sparse Matrix Codes

The basic sparse matrix coding techniques have been described in [1] and [7]. We have found the symbolic and numeric (SYMFAC-NUMFAC) factorization technique to be extremely efficient in solving the linear equations in reservoir simulation. A subroutine SYMFAC is used to generate pointers to the nonzeroes of the triangular factors L and U of the LU factorization of A. Given these pointers, a subroutine NUMFAC is used to factor A (or to convert it to U). A version of SYMFAC-NUMFAC that is particularly well suited for reservoir simulation has been developed here at Yale [5].

For Type 1 linear systems, we apply SYMFAC once per simulation grid and NUMFAC once each time step. In the case of Type 2 linear systems there are two options. First, we can assume that the matrix blocks (see Figure 2) contain only nonzeroes. (The incidence matrix is symmetric as a result.) This allows us to apply SYMFAC once per grid and NUMFAC once per time step as before. Second, we can take into account the zeroes within the matrix blocks at each time step. Then we must apply both SYMFAC and NUMFAC at every time step. The relative advantage of the two options depends on the relative quantities of zeroes and nonzeroes in the matrix blocks and the relative efficiency of the SYMFAC and the NUMFAC subroutines. As a general rule, it would pay to assume that all the matrix blocks are dense for the first time step just to find out the cost of NUMFAC for this case; at subsequent time steps, one would take into account zeroes within the matrix blocks until the combined cost of SYMFAC and NUMFAC exceeded the cost of NUMFAC assuming the dense blocks.

We have made a series of computer timing studies to determine the CPU time requirements of the Yale code when applied to reservoir simulation problems. The computer used for the timing studies was an IBM 370/158 with virtual storage. The timing runs were

made without background jobs to avoid interference. The programs were compiled by using the IBM FORTRAN IV Level H Extended Compiler, with OPT = 2. Selected single precision floating point instruction speeds for the 370/158 are given in Table 4. Table 5 gives the

Instruction	*Average Floating Point Instruction Speed in Microseconds*
Add	2.0
Load	0.70
Multiply	2.0
Store	0.88

Table 4:
Selected Average Instruction Speeds on the IBM 370/158

Grid	*Nonzeroes of U*	*Work*	*SYMFAC*	*NUMFAC*	*Time per Multiply*
30×30	8,915	189,276	.62 sec	1.8 sec	9.4 μsec
50×50	33,961	1,149,772	2.2 sec	10. sec	9.1 μsec
20×60	11,701	232,372	.78 sec	2.2 sec	9.6 μsec
55×72	26,514	709,442	1.8 sec	6.6 sec	9.3 μsec
7×9	913*	11,611	.10 sec	.14 sec	12. μsec
7×9	2,664	51,894	.23 sec	.49 sec	9.4 μsec

Table 5:
CPU Time Requirement of SYMFAC and NUMFAC

CPU time requirement of the Yale codes for several typical cases of Type 1 and Type 2 matrices. In each case, we used the best of the three grid orderings. Note that except for the smallest system, the Yale code requires about 9.5 microseconds per multiplication for NUMFAC, while a band algorithm using the grid row ordering requires about 7.2 microseconds per multiplication. The band algorithm, which does not do any pivoting, is the most efficient of the codes we tested in terms of CPU time per multiplication. But overall the Yale NUMFAC code is considerably faster. For example, it is approximately

* Nonzeroes of L is 1,355.

eight times faster in the case of the 55×72 grid in simulation problem 3. This favorable comparison is due to the reduced number of multiplications (a factor of about ten).

The computer time required to reorder the grid is unimportant relative to the time required to perform the numeric elimination, since in most cases the grid is reordered only once and the system (1) is solved many times in a simulation study. Typically, the computer time to reorder the grid is 0.5 to 4.0 times that required for one numeric elimination.

Storage requirements for sparse elimination are discussed in [4]. It should be noted that the storage required increases very rapidly with increasing grid dimensions. See Table 5 for the actual number of nonzeroes in U for the test problems.

Since we are solving a time-dependent problem and are primarily concerned with speed and not storage, we use a SYMFAC subroutine that returns pointers to the nonzeroes of L and U and a NUMFAC subroutine that returns the numeric values of U only. If we were more concerned about storage (because of hardware limitations), we could use the SYMFAC subroutine described in [9], which returns the "compressed" pointers for the nonzeroes of L and U. Or, going one step further, we could use the TRKSLV subroutine described in [4], which combines the SYMFAC and NUMFAC subroutines and requires storage for only the "compressed" pointers and numeric values for the nonzeroes of U.

Conclusions

1. Sparse Gaussian elimination is a powerful engineering tool that can be used to solve economically many systems of linear equations arising in reservoir simulation.
2. Reordering significantly reduces the number of multiplications required for elimination. For a Type 1 matrix, when a two-dimensional grid is nearly square, the minimum degree ordering is significantly better than the alternate diagonal ordering. The improvement is further enhanced if the grid is irregularly shaped.

3. Sparse Gaussian elimination can be considerably faster than band elimination algorithms. For a 55×72 grid with 2,347 equations, the Yale code reduces computing time by a factor of eight over a band elimination algorithm.

References

1. A. Chang.
Application of sparse matrix methods in electric power analysis.
In R. A. Willoughby, editor, *Sparse Matrix Proceedings*, 113-122. IBM, 1968.

2. S. C. Eisenstat.
Complexity bounds for Gaussian elimination.
To appear.

3. S. C. Eisenstat, M. H. Schultz, and A. H. Sherman.
Applications of an element model for Gaussian elimination.
Symposium on Sparse Matrix Computations, Argonne National Laboratory, 1975.

4. S. C. Eisenstat, M. H. Schultz, and A. H. Sherman.
Considerations in the design of software for sparse Gaussian elimination.
Symposium on Sparse Matrix Computations, Argonne National Laboratory, 1975.

5. S. C. Eisenstat, M. H. Schultz, and A. H. Sherman.
A user's guide to the Yale sparse matrix package.
To appear.

6. J. A. George.
Nested dissection of a regular finite element mesh.
SIAM Journal of Numerical Analysis 10:345-363, 1973.

7. F. G. Gustavson.
Some basic techniques for solving sparse systems of linear equations.
In D. J. Rose and R. A. Willoughby, editors, *Sparse Matrices and Their Applications*, 41-52. Plenum Press, 1972.

8. H. S. Price and K. H. Coats.
Direct methods in reservoir simulation.
Society of Petroleum Engineers Journal 14:295-308, 1974.

9. A. H. Sherman.
On the Efficient Solution of Sparse Systems of Linear and Nonlinear Equations.
PhD thesis, Yale, 1975.

10. P. T. Woo, S. J. Roberts, and F. G. Gustavson.
Application of sparse matrix techniques in reservoir simulation.
SPE 4544, 48th Annual Fall Meeting of the Society of Petroleum Engineers, Las Vegas, Nevada, 1973.